app.py

```python
from utils import *

from object import Vertex, Shape2D, Shape3D, Object

class App:
    """
    Main class of the project. Initilising window, lights, manages objects and draw them.
    """
    def __init__(self, _resolution : tuple[int, int] = DEFAULT_RESOLUTION) -> None:
        pygame.init()
        pygame.display.set_caption('Plato Solids')
        try:
            pygame.display.set_icon(pygame.image.load('./rsc/icosahedron.png'))
        except:
            try:
                pygame.display.set_icon(pygame.image.load('./icosahedron.png'))
            except:
                pass
        self.display = _resolution
        self.window = pygame.display.set_mode(self.display, DOUBLEBUF | OPENGL)
        gluPerspective(45, (self.display[0] / self.display[1]), 0.1, 50.0)
        glTranslatef(*CAM_POSITION)

        glLight(GL_LIGHT0, GL_POSITION,  (4, 4, 8, 1)) # point light from the left, top, front
        glLightfv(GL_LIGHT0, GL_AMBIENT, (0, 0, 0, 1))
        glLightfv(GL_LIGHT0, GL_DIFFUSE, (1, 1, 1, 1))

        glEnable(GL_DEPTH_TEST)
        # glDepthFunc(GL_LESS)

        self.SHAPES_DEBUG : list[Object] = list()
        self.SHAPES : list[Object] = list()

    def add_debug_object(self, _object : Object) -> None:
        """
        Adds new objects, that are used as debug .
        """
        self.SHAPES_DEBUG.append(_object)

    def add_object(self, _object : Object) -> None:
        """
        Adds new objects, that will be drawn.
        """
        self.SHAPES.append(_object)

    def draw_objects(self, _debug):
        """
        It draws all the objects you added .
        """

        if _debug:
            for shape in self.SHAPES_DEBUG:
                shape.draw(True, False)

        for shape in self.SHAPES:

            shape.draw(True, True)
            shape.translate()
            shape.rotate()
            shape.collide()

            for _shape in self.SHAPES:
                if shape is not _shape:
                    # print()
                    def lst_mul(lst, x):
                        return [a * x for a in lst]
                    if shape.position.distance(_shape.position) <= (shape.edge +
_shape.edge) * .7:
                        shape.translation, _shape.translation = lst_mul(shape.translation,
-1), lst_mul(_shape.translation, -1)

            # print(shape)

    def run(self, _debug : bool = False):
        """
        Runs project. You can rotate cam Up/Down and Left/Right.
        """
        clock = pygame.time.Clock()
        is_over = False

        global FPS_COUNT

        while not is_over:
            dX = 0
            dY = 0
            dZ = 0
            angle = 0
            for event in pygame.event.get():
                if event.type == pygame.QUIT or event.type == pygame.K_ESCAPE:
                    is_over = True
                    pygame.quit()
                    quit()
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_UP:
                        dX += 1
                        angle = 15
                    if event.key == pygame.K_DOWN:
                        dX -= 1
                        angle = 15

                    if event.key == pygame.K_LEFT:
                        dY += 1
                        angle = 15
                    if event.key == pygame.K_RIGHT:
                        dY -= 1
                        angle = 15

                    if event.key == pygame.K_KP7:
                        dZ += 1
                        angle = 15
                    if event.key == pygame.K_KP9:
                        dZ -= 1
                        angle = 15

            glRotatef(angle, dX, dY, dZ)
            # glRotatef(.5, 1, 1, 1)

            FPS_COUNT += 1

            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

            glEnable(GL_LIGHT0)
            glEnable(GL_LIGHTING)
            glEnable(GL_COLOR_MATERIAL)

            self.draw_objects(_debug)

            glDisable(GL_LIGHT0)
            glDisable(GL_LIGHTING)
            glDisable(GL_COLOR_MATERIAL)

            clock.tick(FPS_CAP)
            # pygame.display.set_caption(f'{int(clock.get_fps())} [{FPS_COUNT}]')
            pygame.display.flip()

            # pygame.time.wait(30)
            # sleep(2)

if __name__ == '__main__':
    pass
```